29 November 2025 10:06

Priority Queue

Queue => FIFO => First In First out

Priority Queue => highest priority

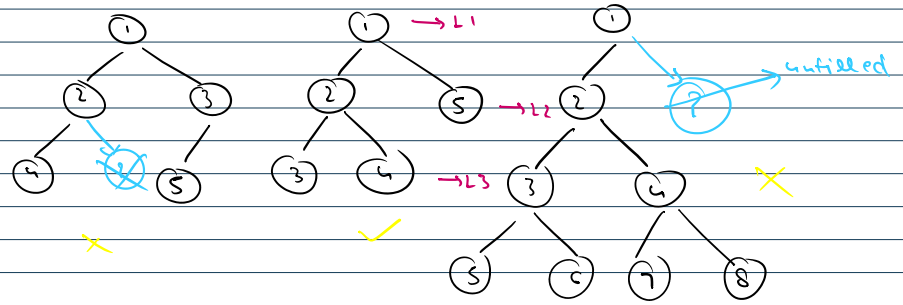
↓ Implement

Heap + Complete Binary Tree

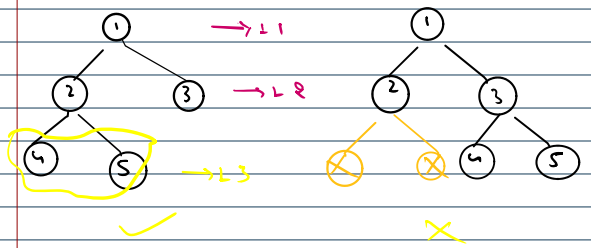
Priority Queue
- → min PQ => min element
- → max PQ => max element

PQ
- → add
- → remove
- => size
- peek
- empty

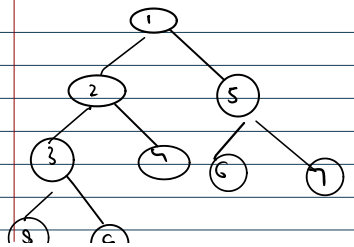
Complete Binary Tree



1. all levels are completely filled (except last)
2. last level filled left to right.



Complete Binary Tree

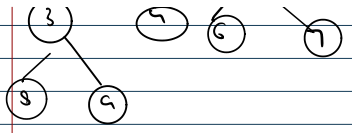


arr => {1, 2, 5, 3, 4, 6, 7, 8, 9}
0 1 2 3 4 5 6 7 8

4 => parent = 2 (4 → 1)

7 => parent = 5 (6 → 2)

01 => parent = 3 (8 → 3)



{i} => parent = (i-1)/2

5 => child → lc = 6 → 5
→ rc = 7 → 6

3 => child → lc = 8 → 7
→ rc = 9 → 8

{i} => child → left = 2i+1
→ right = 2i+2

Heap

A heap is a CBT that satisfies the heap property

maxHeap => root = max element
minHeap => root = min element

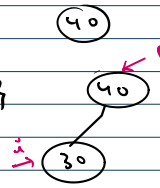
maxHeap

arr → {40, 30, 50, 10, 20, 60, 70, 5, 15, 25}

h => { }

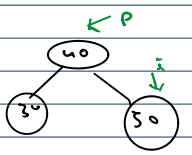
40 => h => {40}

30 => h => {40, 30}



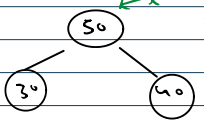
i=1 => p = (i-1)/2 = 0
h[i] > h[p] => swap (float up)
30 > 40 → x (No swap)

50 => h => {40, 30, 50}



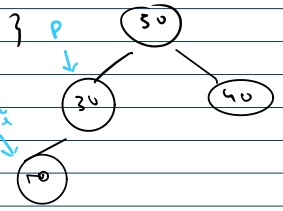
i=2 => p = (i-1)/2 = 0.5 = 0
h[i] > h[p] => swap (float up)
50 > 40 → ✓ (swap)

{50, 30, 40}



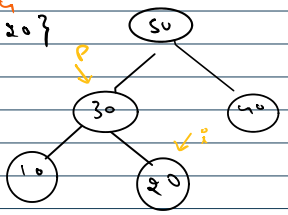
=> recall float up again
!!
x

10 => h => {50, 30, 40, 10}

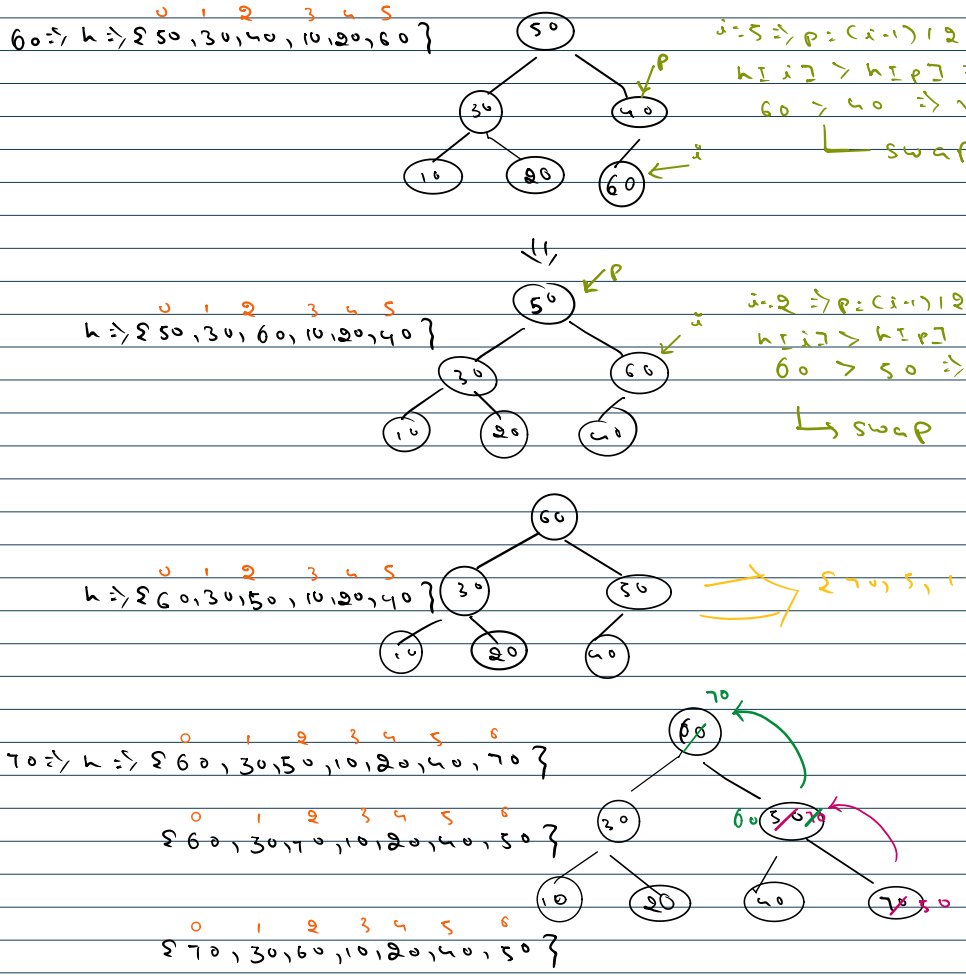


i=3 => p = (i-1)/2 = 1
h[i] > h[p] => ?
10 > 30 => x => No swap

20 => h => {50, 30, 40, 10, 20}



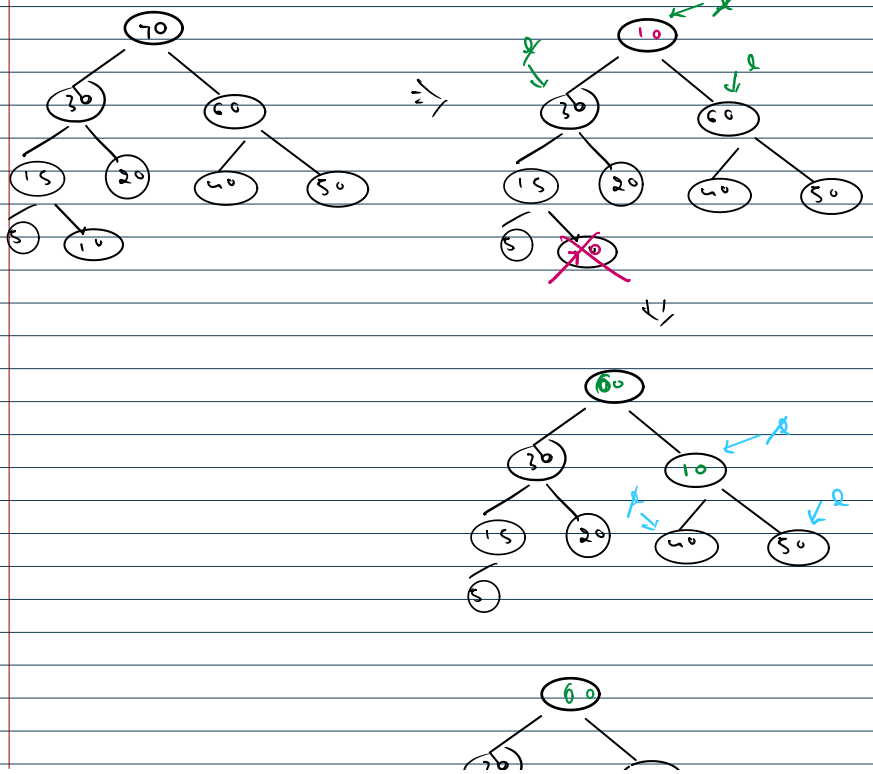
i=4 => p = (i-1)/2 = 1.5 = 1
h[i] > h[p] => ?
20 > 30 => x => No swap

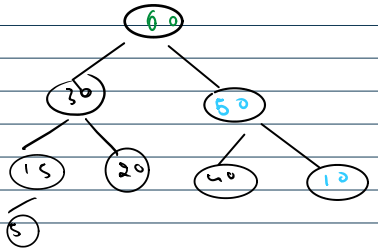


Removal from Heap

h => { 70, 30, 60, 15, 20, 40, 50, 5, 10 }

h => { 10, 30, 60, 15, 20, 40, 50, 5, ~~70~~ }





Remove again